PingLess Play

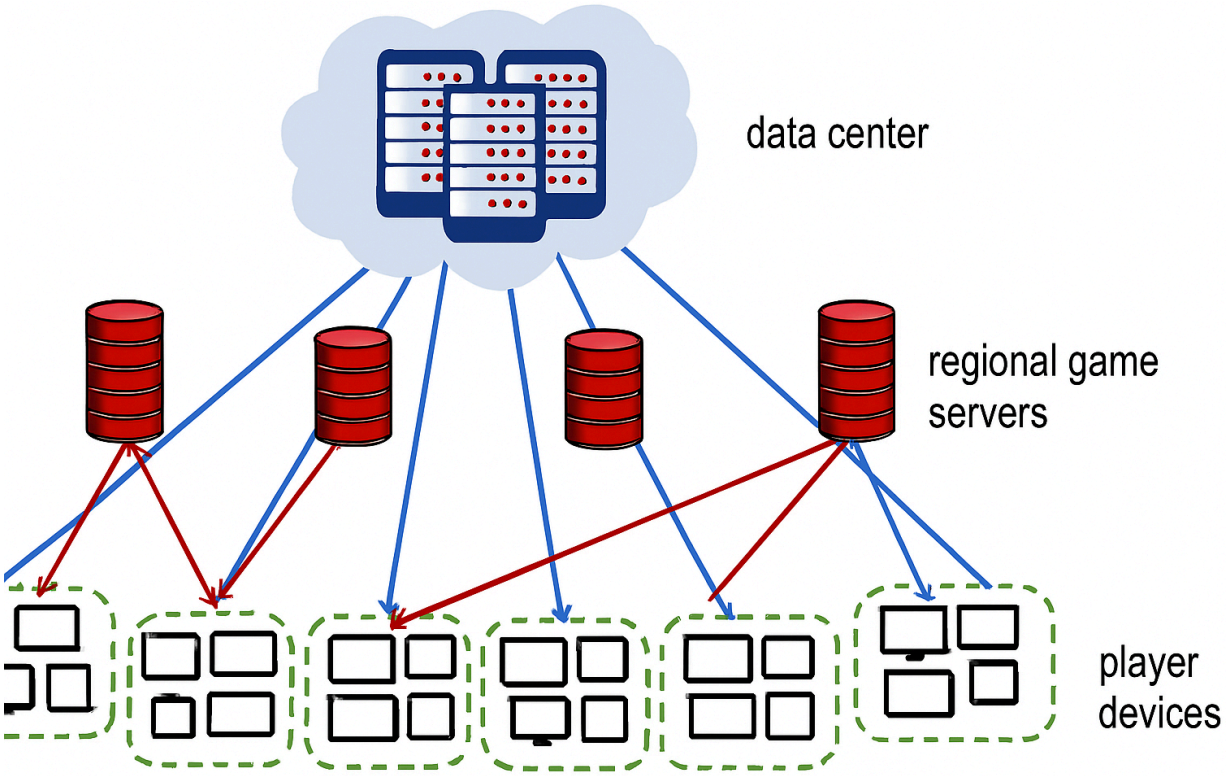
Ever wondered how cloud gaming platforms like NVIDIA GeForce NOW or Xbox Cloud Gaming deliver smooth, lag-free gameplay to users across the globe? As games become more graphics-intensive and player bases grow, it's critical that the infrastructure behind these platforms must be optimized to provide real-time experience.

To achieve this, gaming providers deploy **regional game servers** that store and stream popular game assets. When a player launches a game, the system tries to serve it from a nearby server rather than a distant data center, reducing latency and improving responsiveness.

But here's the challenge: Which game assets should be stored in which regional servers to minimize the average latency across all player sessions?

Problem:

Given a description of regional game servers, player regions, and game assets, along with predicted demand for individual assets, decide which assets to store in which servers in order to minimize the average latency for all player sessions.



Game Assets

Each game asset (e.g., map, texture pack, character model) has a size in megabytes (MB). The central game repository stores all assets. Each asset can be stored in 0,1 or multiple **regional servers**. Each regional server has a maximum capacity in MB.

Player Regions

Each **region** represents a group of players connecting from the same geographical area (e.g., a city or neighborhood). All regions are connected to the **central repository**. Each region may be connected to one or more regional game servers.

Each region has 2 characteristics

- **Latency to the central repository** (how long it takes to stream an asset from the central server).
- **Latencies to each connected regional server** (how long it takes to stream an asset from that server).

Requests

The predicted request data indicates how many times a particular game asset is requested from a particular region. This helps prioritize which assets should be stored closer to which players.

Input Data Format

The input is provided as a plain text dataset file using only ASCII characters, with UNIX-style line endings (`\n`).

Game assets, player regions, and game servers are referenced by **integer IDs**:

- **A** assets numbered from `0` to `A-1`
- **P** player regions numbered from `0` to `P-1`
- **G** game servers numbered from `0` to `G-1`

File Format

First line of input contains five numbers:

- `A` – Number of game assets (1 ≤ A ≤ 10,000)

- `P` – Number of player regions (1 ≤ P ≤ 1,000)
- `R` – Number of request descriptions (1 ≤ R ≤ 1,000,000)
- `G` – Number of regional game servers (1 ≤ G ≤ 1,000)
- `X` – Capacity of each game server in MB (1 ≤ X ≤ 500,000)

Second line contains `A` numbers:

- S_0 S_1 ... S_{a-1} – Sizes of individual game assets in MB ($1 \le S_i \le 1{,}000$)

The next section describes the player region one after the other. For each region `0` to `P-1`, the following lines appear:

- A line with two numbers:
  - `LD` – Latency from central repository to this region (200 ≤ LD ≤ 4000 ms)
  - `K` – Number of regional servers connected to this region (0 ≤ K ≤ G)
- Then `K` lines, describing the connections from the player region to each of the K connected regional servers. Each line contains the following numbers:
  - `g` – ID of the connected game server (0 ≤ g < G)
  - `Lr` – Latency from this server to the region in milliseconds (1 ≤ Lr ≤ 500 ms, and Lr < LD)

The last section contains R requests. Each of the `R` lines contains:

- R_a – ID of the requested game asset ($0 \le R_a < A$)
- R_p – ID of the player region making the request ($0 \le R_p < P$)
- R_n – Number of requests from that region for that asset ($0 < R_n < 10000$)

Example Input

```
5 2 4 3 100
50 50 80 30 110
1000 3
0 100
2 200
1 300
500 0
3 0 1500
0 1 1000
4 0 500
1 0 1000
```

5 game assets, 2 player regions, 4 request descriptions, 3 regional servers (100MB each).
Assets 0–4 have sizes: 50MB, 50MB, 80MB, 30MB, 110MB.
Region 0:

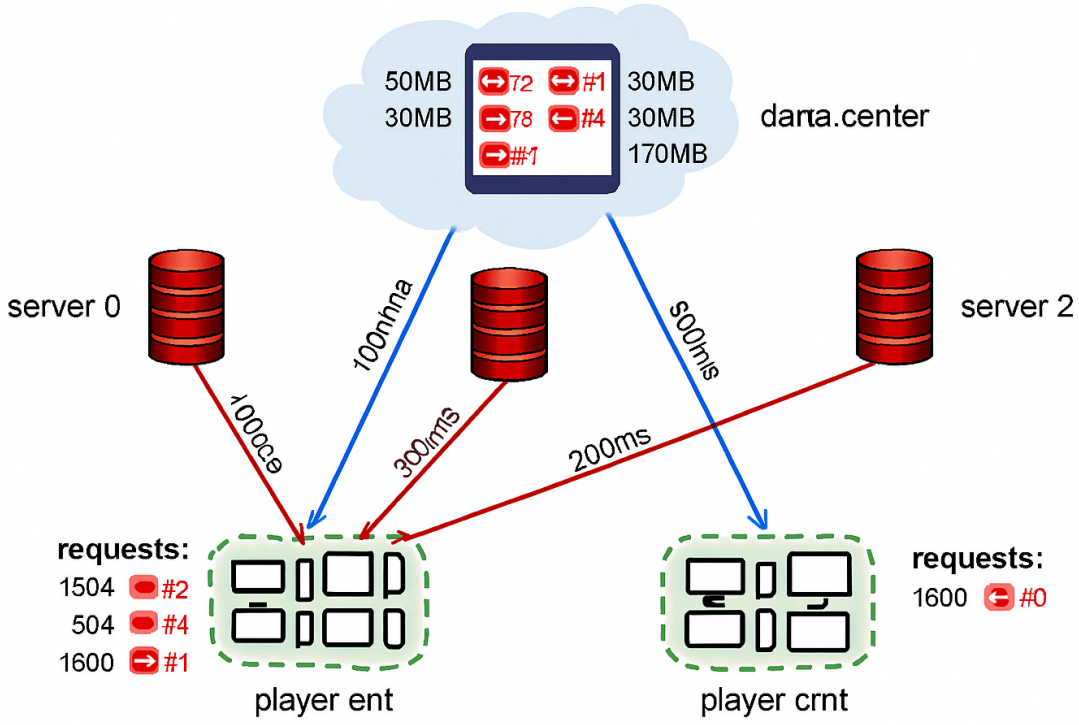
- 1000ms latency to central server
- Connected to 3 regional servers:
  - Server 0: 100ms latency
  - Server 2: 200ms latency
  - Server 1: 300ms latency

Region 1:

- 500ms latency to central server
- No regional server connections

Requests:

- 1500 requests for asset 3 from region 0
- 1000 requests for asset 0 from region 1
- 500 requests for asset 4 from region 0
- 1000 requests for asset 1 from region



Submission Format

Your submission should begin with a line containing a single number **N** (where `0 ≤ N ≤ G`) — the number of regional game server descriptions to follow.

Each of the subsequent **N** lines should describe the game assets stored in a single regional server. Each line must contain:

- `g` – the ID of the regional game server ($0 \le g < G$)
- a_0 a_1 ... a – the IDs of the game assets stored in this server ($0 \le a_i < A$), listed in any order without duplicates

Notes:

- Each server should be described in at most one line.
- It is not necessary to describe all servers. If a server is not mentioned, it is considered empty.
- Servers can be described in any order.

Example Submission File

```
3
0 2
1 3 1
2 0 1
```

- We are using all 3 regional servers.
- Server 0 contains only asset 2.
- Server 1 contains assets 3 and 1.
- Server 2 contains assets 0 and 1.

Validation Criteria

The output file is valid if:

- The format matches the specification above.
- The total size of game assets stored in each server does not exceed its maximum capacity.

Scoring Mechanism

The score reflects the average latency saved per request, measured in microseconds. (Inputs are in milliseconds; scoring uses microseconds for better resolution.)

For each request description in the input file, the system selects the best way to stream the asset R_a to the region R_p, based on the lowest possible latency:

- LD – latency from the central repository to region R_p
- $Lc_0, Lc_1, \ldots, Lc_{-1}$ – latencies from connected regional servers to region R_p that:
    - Are connected to R_p
    - Contain the requested asset R_a

The **time saved per request** is calculated as LD − L, where L is the lowest latency among eligible sources. If the asset is streamed from the central repository, the time saved is 0.
For a request description involving R_n requests, the total time saved is: $R_n \times (LD - L)$

Final Score

To compute the total score for the dataset:

1. Sum the time saved across all request descriptions (in milliseconds).
2. Multiply the result by 1000 to convert to microseconds.
3. Divide by the total number of requests across all descriptions.
4. Round down to the nearest integer.

Example Scoring Breakdown

Let's say region 0 has three request descriptions:

- 1500 requests for asset 3, streamed from server 1 with 300ms latency, saves 1000ms - 300ms = 700ms per request
- 500 requests for asset 4, streamed from the central repository, saves 0ms
- 1000 requests for asset 1, streamed from server 2 with 200ms latency, saves 800ms per request

Region 1 has:

- 1000 requests for asset 0, streamed from the central repository → saves 0ms

Total time saved = $(1500 \times 700 + 500 \times 0 + 1000 \times 800 + 1000 \times 0) = 1{,}850{,}000$ ms

Total requests = $1500 + 500 + 1000 + 1000 = 4000$

Average time saved per request = $(1{,}850{,}000/4000) * 1000 = 462{,}500$ µs